Christopher Vajdic
770527-8799
732A38 - Lab1

Assignment 1

The following code was run in R.

```
> x1<-1/3;
> x2<-1/4;
> if (x1-x2==1/12){
+ print("Teacher said true")
+ } else{
+ print("Teacher lied")}
[1] "Teacher lied"
```

This results illustrates a common error in using floating point arithmetic. Division on a non-dense number system can result in rounding towards the nearest neighbor in the case of a non-terminating expansion.

The distance between two representable numbers of the same exponent *e* for a given base *b* and fractional digits *p* is given by *b^e-p*. In practice, the given level of precision is determined by the specific data type used to represent the variable.

The rounding scheme used by floating point numbers is nearest neighbor with a tie decided by "round to even". The relative error resulting from such rounding to a value of unity can be expressed as δ/(1+δ), where δ is the distance rounded off. The absolute magnitude of this value is defined as the *machine epsilon (macheps)*. The machine epsilon is given by the formula

$$\varepsilon = \tfrac{1}{2} b^{-p}.$$

For data type *float* (binary32), the available bits are segmented as sign: 1 bit, exponent: 8 bit, and significand: 23 bit. Therefore the epsilon is 2^-24. For data type *double* (binary64), the available bits are segmented as sign: 1 bit, exponent: 11 bit, and significand: 52 bit. Therefore the epsilon is 2^-53.

The previous code can then be modified so equality is represented by an interval with the offset on either side given by the machine epsilon for the given data type.

```
 > e<-2^-24;
> if (x1-x2<1/12+e && x1-x2>1/12-e){
+ print("Teacher said true")
+ } else{
+ print("Teacher lied")}
[1] "Teacher said true"
```

With the proposed modification, the code produced the expected result that 1/3 − 1/4 = 1/12. Choosing an overly large machine epsilon would lead to "false positives" for the equality operator and conversely, too small of a value for epsilon can provide "false negatives" as exhibited in the original coding.

Assignment 2

A formula for calculating the derivative at a given point x=a is

$$f'(a) = (f(a + e) - f(a))/e$$

, which gives the derivative as $e \to 0$. The following output was generated in R by implementing a function named "deriv" based on the formula above and evaluating the constant function f(x) = x at x = 100000.

```
> deriv <- function(y)
+ {
+ e <- 10^-15
+ result=((y+e)-y)/e
+ }
> attributes(deriv);
$source
[1] "function(y)"        "{"                 "e <- 10^-15 "
[4] "result=((y+e)-y)/e" "}"


> deriv(100000);
> a <- deriv(100000);
> a;
[1] 0
```

The resulting value is obviously incorrect as the slope of the constant function is equal to unity. The erroneous output is due to the fact the chosen epsilon value is smaller than the machine epsilon. The chosen epsilon was 10^-15 = 2 ^-49.82892. Therefore, f(x + e) = f(x) with a resulting value of zero in the numerator.

Assignment 3

The formula for calculating the variance of a random variable is given as

$$\mathrm{Var}(x) = 1/(n - 1)(\sum x_i^2 - 1/n(\sum x_i)^2).$$

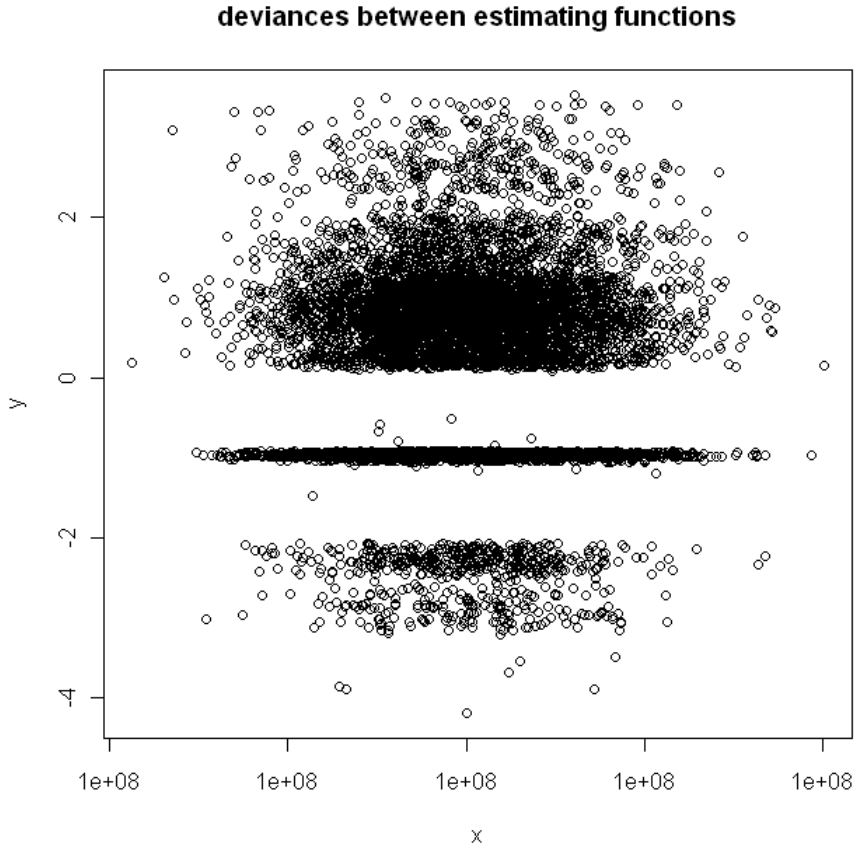
A function *myvar* was implemented in R based on the above formula and compared to the R base package's own variance calculating function *var* on a sample of 10000 normally distributed random variables ~N(10^8,1). The following code illustrates the procedure used.

```
myvar <- function(y)
{
```

```
n <- length(y)
result <-  (sum(y^2) - (sum(y)^2)/n)/(n - 1)
}

x <- rnorm(10000, 10^8, 1)
y<-1:10000;
for(n in 1:10000) (
y[n] <-  myvar(x[1:n]) - var(x[1:n])
)
win.graph()
plot(x,y,main="deviances between estimating functions");
```

The following scatterplot explores the relationship between the random vector values on the horizontal axis and the difference in variance estimating functions on the vertical.



A clear horizontal banding occurs which indicates the non-dense nature of floating point arithmetic. Due to rounding, the deviances between the two functions gravitate towards the representable numbers for the data types with variation a result of the initial

distribution created by the *rnorm* function. *Myvar* shows a bias towards overestimation of the variance in comparison to the *var*.